Amendments to the Specification:

Please replace the paragraph beginning on page 8, line 32, with the following rewritten paragraph:

The computing system 100 also comprises several software elements. In the illustrated embodiment, these software elements include:

- the application 175, cache 185, and VMS 190 residing on the computing device 110;
- the file system 180 and the cache 187 residing on the server 120; and
- the disk storage 195 residing on the disk storage unit 125.

As with the hardware, the invention admits a great deal of variation in the implementation of the software elements 175, 180, 185, 187, 190, and 195. For instance, the cache 185 will typically be a least recently used queue as is known in the art, although other types of data structures might be used.

Please replace the paragraph beginning on page 10, line 28, with the following rewritten paragraph:

Thus, in a typical scenario, the client 110 will request a page of data 105 from the server 120. The server 120 then sends the requested page of data 105, which is in the clean state 320, to the client 110. The client 110 buffers the clean data in the cache 185~~180~~, and the VMS 190 enters it onto the clean list (not shown) for the client 110. The client 110 modifies the data in the page, whereupon the page transitions 360 to the dirty state 310 and the VMS 190 moves the page to the dirty list (not shown). At some point, the client 110 writes the page of data 105 to the server ~~cache~~ 120, and the page of data 105 transitions 340 to the written state 330. The server 120 buffers the page of data 105 in the cache 187. Note that the client 110 continues to consider the page of data 105 to be in the written state 330 while it is buffered by the server 120. The server 120 eventually writes the page of data 105 to the disk storage 125. The client system_110 eventually sends a file sync request to the server I20. The page of data 105 at the client 110 transitions 350 to the clean state and the VMS 190 moves the page of data 105 back to the clean list.

Please replace the paragraph beginning on page 15, line 6, with the following rewritten paragraph:

FIG. 6 illustrates the computing system 100a of the cluster 405a in greater detail. One feature of this implementation is the cache 185b. In this particular embodiment, dirty pages l05a at the client computing device 110a may be buffered before being written to the server 120a. The buffered dirty pages 105a are then written at a convenient, economical point in the operation of the client computing device 110a. Thus, efficiency may be gained at two points in this particular embodiment—routine writes from the client-computing device 110a to the server 120a and routine writes from the server 120a to the disk storage unit 125a. Of course, there may be circumstances where such efficiencies are not realized because the computing system 100a cannot wait for the routine writes. On the whole, however, this particular implementation has demonstrated significant increases in efficiency.

Please replace the paragraph beginning on page 15, line 17, with the following rewritten paragraph:

Note that the sequence number can change in other contexts. For instance, the CFS will be mounted to a new server if the server on which it is originally mounted fails for some reason. The new server will already be booted, and need not be rebooted to effect the mount. However, the re-mount of the CFS on the new server will modify the UID assigned in the box 704 of Fig. 7A.

Please replace the paragraph beginning on page 16, line 1, with the following rewritten paragraph:

The computing system 100a then begins "routine" operations, as set forth in the box 520. "Routine" in this context means that the application's execution does not call for the reclamation of a dirty or written page. The conduct of routine operations is more thoroughly illustrated in FIG. 7B. The application software 175a begins routine operations by reading pages 105a from the server 120a, as set forth in the box 705~~708~~. As shown in FIG. 3, if the application modifies the data, the page l05a transitions 360 from a clean state 320 to the dirty state 310. Returning to FIGS. 6 and 7B, the application software 175a buffers the pages l05a as it uses them in the cache 185b on the client computing device 110a.

Please replace the paragraph beginning on page 18, line 1, with the following rewritten paragraph:

Returning to FIG. 5, the computing system 100a may then return to routine operations as set forth in the box 520, attempt to claim a dirty page again as set forth in the box 530, or attempt to claim a written page as set forth in the box 540. When the client computing system 110a requests a written page, as set forth in the box 726, the client 110a compares the current UID and the buffered UID as set forth in the box 728. The client 110a then determines whether the buffered UID and the current UID match, as set forth in the box 730.

Please replace the paragraph beginning on page 18, line 8, with the following rewritten paragraph:

The client 110 then handles the request depending upon the outcome of the determination in the box 730. If the buffered UID and the current UID are different, then the VMS 190a and FS 180a will know there has been a server failure. The written page 105a is then rewritten from the client computing device 110a to the server 120a set forth in the box 732. Once the page 105a is rewritten to the server 120a, or if the buffered UID and the current UID match, then a file sync operation is performed so that the written page 105a is written to the disk storage 190a, as set forth in the box 734. Once the file sync op is completed, the page ~~105~~ 105a transitions to the clean state and can be reclaimed. The page l05a can then be claimed as set forth in the box 736.

Please replace the paragraph beginning on page 30, line 2, with the following rewritten paragraph:

A method and apparatus for delaying asynchronous writes in a distributed file system, wherein the file system includes a unique identifier ("UID") ~~is~~

~~disclosed~~. The method comprises buffering a page of dirty data with the unique identifier upon writing to the server; changing the unique identifier ("UID") to create a current unique identifier upon a failure of the server; comparing the buffered unique identifier with the current unique identifier when the page is requested while the page is in a written state; and handling the request responsive to the comparison. In other aspects of the invention, the invention comprises a computer programmed to perform the method and a program storage medium encoded with instructions that, when executed by a computer, perform the method.